AD

TECHNICAL REPORT ARCCB-TR-95016

A MATHEMATICA FORMULATION OF GEOMETRIC ALGEBRA IN 3-SPACE

L.V. MEISEL

MARCH 1995

US ARMY ARMAMENT RESEARCH,
DEVELOPMENT AND ENGINEERING CENTER
CLOSE COMBAT ARMAMENTS CENTER
BENÉT LABORATORIES
WATERVLIET, N.Y. 12189-4050

DISCLAIMER

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The use of trade name(s) and/or manufacturer(s) does not constitute an official indorsement or approval.

DESTRUCTION NOTICE

For classified documents, follow the procedures in DoD 5200.22-M, Industrial Security Manual, Section II-19 or DoD 5200.1-R, Information Security Program Regulation, Chapter IX.

For unclassified, limited documents, destroy by any method that will prevent disclosure of contents or reconstruction of the document.

For unclassified, unlimited documents, destroy when the report is no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
March 1995

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
A MATHEMATICA FORMULATION OF GEOMETRIC ALGEBRA IN 3-SPACE

5. FUNDING NUMBERS
AMCMS: 6111.02.H611.1

6. AUTHOR(S)
L.V. Meisel

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army ARDEC
Benét Laboratories, AMSTA-AR-CCB-O
Watervliet, NY 12189-4050

8. PERFORMING ORGANIZATION REPORT NUMBER
ARCCB-TR-95016

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army ARDEC
Close Combat Armaments Center
Picatinny Arsenal, NJ 07806-5000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Published in: *American Journal of Physics*

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)
This report, in conjunction with its multivector analysis package, provides: (1) a brief introduction to the ideas and features of the 8-dimensional geometric algebra G(3) defined on 3-space; (2) a code for performing geometric algebra analysis; (3) examples of the operation of the code; and (4) applications of geometric algebra to the solution of multivector equations and to rotation operations in 3-space.

14. SUBJECT TERMS
Mathematica, Geometric Algebra, Multivector Algebra

15. NUMBER OF PAGES
36

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

TABLE OF CONTENTS

1.0 INTRODUCTION 1
2.0 THEORETICAL BACKGROUND 2
  2.1 Dot and Wedge Products 3
  2.2 Geometric Products of Multivectors With Scalars and Pseudoscalars 3
  2.3 Products of Vectors With Vectors in G(3) 4
  2.4 Products Involving Bivectors (Pseudovectors) in G(3) 5
  2.5 The Algebra of G(3) 6
3.0 EXAMPLES 6
  3.1 Dot, Cross, and Bivector Products 7
  3.2 Check on Associativity of Geometric Products 9
  3.3 Elementary Properties of Involuntary Transformations of Products 10
  3.4 Inverse of General Multivectors 10
4.0 APPLICATIONS 12
  4.1 Solution of Multivector Equations 12
  4.2 Rotation Operators in 3-Space 19
REFERENCES 29
APPENDIX: THE PACKAGE 30

1.0 INTRODUCTION

David Hestenes (ref 1) presented a definitive formulation of the geometric algebra G(3) for problems in 3-space R(3) with applications to mechanics. Reference 1 also presents a historical review of the development of a geometric algebra and collects many previously published results (ref 2).

Baylis, Huschilt, and Wei (ref 3) presented a dissertation on geometric algebra in 3-space without wedge products. They accomplished this by using the dual relation between the wedge product and the cross product of Gibbs vector analysis:

$$<a>\wedge<b> = i\ <a>X<b>, \tag{1}$$

where i is the unit pseudoscalar, the Gibbs cross product $<a>X<b>$ is the Hodge dual of the wedge product $<a>\wedge<b>$, and we denote vectors by angular brackets. Reference 3 also carried over the Gibbs dot product, etc. Thus, Baylis, Huschilt, and Wei (ref 3) were able to define geometric products of general elements, multivectors, of G(3) by assuming the results of Gibbs vector analysis, identifying bivectors (pseudovectors) with cross products, and enumerating certain features of the pseudoscalar i.

The approach of Reference 3 obviates the need for enumerating the properties of wedge products, which are simply inherited from the properties of cross products, etc. Furthermore, it turns out that all the properties of the geometric product in G(3) may be expressed in a simple way in terms of Gibbs dot and cross products. Of course, when one builds the features of Gibbs vector analysis into the geometric

product, one cannot show (as is done in Reference 1) how identities in Gibbs vector analysis follow from properties (e.g., associativity) of the geometric product.

In this report, the methods of Baylis, Huschilt, and Wei for geometric algebra in R(3) are implemented. The report is organized as follows: Section 2, Theoretical Background, reviews the properties of sums and products of the elements of the 8-dimensional algebra G(3). Section 3, Examples, demonstrates standard features of geometric products and defines inverses for general multivectors in G(3) using the package. Section 4, Applications, demonstrates the utility of the geometric algebra code for the solution of multivector equations and to rotation operations in 3-space. The Appendix contains the MV package, which defines data types "MV" for multivectors and "vec" for vectors, and presents a code for (1) performing standard Gibbs vector analysis; (2) the geometric product in G(3); and (3) special coordinate specific (vectors in list form) calculations.

2.0 THEORETICAL BACKGROUND

A general element of a geometric algebra is referred to as a multivector. A multivector $\mathbf{m}$ in G(3) is generally a sum of four parts,

$$\mathbf{m} = a_0 + \mathbf{a} + i\ \mathbf{A} + i\ A_0, \tag{2}$$

where a_0 and A_0 are scalars, and $\mathbf{a}$ and $\mathbf{A}$ are vectors. The scalar part of $\mathbf{m}$ (a_0) is said to be of grade 0, the vector part of $\mathbf{m}$ ($\mathbf{a}$) of grade 1, the bivector part of $\mathbf{m}$ ($i\ \mathbf{A}$) of grade 2, and the

pseudoscalar part of **m** (i A_0) of grade 3. Terms of higher grade vanish in G(3). The vector **A** is said to be the dual of the bivector part (i **A**). The unit pseudoscalar i is of grade 3, it commutes with all multivectors, its square is -1, and it has geometrical content; it is not a complex number. Scalars and vectors are defined over R(1) and R(3), respectively. Hestenes refers to pure grade r multivectors as r-blades (e.g., pure vectors are referred to as 1-blades).

As discussed in References 1 through 3, the r-blade parts of multivectors have geometrical significance. One can associate vectors (1-blades) with directed lines, bivectors (2-blades) with oriented areas, and pseudoscalars (3-blades) with oriented volumes.

2.1 Dot and Wedge Products

In this report, the wedge product of vectors is defined in terms of the cross product through Eq. (1), and the dot product is carried over directly from Gibbs vector analysis. In general, for q≥r, the dot product of a q-blade with an r-blade is the grade (q-r)-part of the geometric product of the blades and the wedge product is the grade (q+r)-part of the geometric product of the blades. Although we include a discussion of wedge products in this work, it is not incorporated into the package.

2.2 Geometric Products of Multivectors With Scalars and Pseudoscalars

Scalars and pseudoscalars commute with all multivectors. Pseudoscalar geometric products change grade and are analogous to

multiplication by imaginary numbers, and scalar geometric products are analogous to multiplication by reals on C(1). Geometric products with a pseudoscalar are always equal to dot products, and the product of an r-blade with a pseudoscalar is a grade (3-r) blade. Thus,

$$c\ m = m\ c, \tag{3}$$

$$(c\ i)\ m = m\ (c\ i) = c\ m.i = c\ i.m, \tag{4}$$

where m is an arbitrary multivector, c is an arbitrary scalar, (c i) is an arbitrary pseudoscalar, and i is the unit pseudoscalar.

2.3 Products of Vectors With Vectors in G(3)

The geometric product of vectors in G(3) is defined as

$$<a><b> = <a>.<b> + i\ <a>X<b> = (a,b) + i\ <a>X<b>, \tag{5}$$

where the grade 0 part of the product <a>.<b> = (a,b) is the Gibbs scalar (or dot) product of <a> and <b>, and the grade 2 part of the product is i(<a>X<b>) where <a>X<b> is the Gibbs vector (or cross) product of <a> and <b> (a vector quantity). Employing the properties of the dot and cross products we see that

$$<b><a> = <b>.<a> + i\ <b>X<a> = (a,b) - i\ <a>X<b> \tag{5a}$$

which implies that

$$(a,b) = <a>.<b> = (<a><b> + <b><a>)/2 \tag{5b}$$

and $$i\ <a>X<b> = <a>\wedge<b> = (<a><b> - <b><a>)/2. \tag{5c}$$

From the properties of the Gibbs dot and cross, the scalar

$$<a><a> = (a,a) > 0, \tag{5d}$$

if <a> is not a zero vector.

2.4 Products Involving Bivectors (Pseudovectors) in G(3)

As indicated in Eq. (2), a general bivector can be expressed as (i A) where A is a vector (the Hodge dual of i A). Thus, since i commutes with all multivectors, the properties of geometric products of bivectors can be deduced from those of products involving vectors.

2.4.1 Product of a Vector and a Bivector in G(3)

(i<A>)<b> = i <A><b> = i (<A>.<b> + i <A>X<b>)

= i <A>.<b> - <A>X<b> = i (A,b) - <A>X<b>. (6a)

We see that the geometric product of vectors with bivectors yields the sum of a pseudoscalar and a vector part. Forming <b>(i<A>) and using the properties of Gibbs dot and cross products as with Eq. (5), we can pick off the dot and wedge products:

(i<A>).<b> = -<A>X<b> = ((i<A>)<b> - <b>(i<A>))/2 (6b)

and (i<A>)∧<b> = i(<A>.<b>)= ((i<A>)<b> + <b>(i<A>))/2. (6c)

2.4.2 Product of Bivectors in G(3)

(i<A>)(i<B>) = -<A><B> = -(<A>.<B> + i<A>X<B>) (7a)

We see that the geometric product of bivectors is equal to the negative of the product of their dual vectors. We can pick off the dot and wedge products:

(i<A>).(i<B>) = -<A>.<B>

= ((i<A>)(i<B>) + (i<B>)(i<A>))/2 (7b)

and (i<A>)∧(i<B>) = 0. (7c)

The bivector part (which is neither dot nor wedge) is seen to be given by the commutator product of (i<A>) and (i<B>). To every

plane, one can associate a "unit" bivector, which is a square root of -1; its Hodge dual is a directed normal to the plane.

2.5 The Algebra of G(3)

Beyond the properties described above, the algebra has the following features: addition of multivectors is commutative, distributive, and associative. The geometric product of multivectors is distributive and associative. There exist unique multivectors 0 and 1, which serve as identity elements for addition and geometric products, respectively, i.e.,

$$\mathbf{m} + 0 = \mathbf{m} \text{ and } 1\,\mathbf{m} = \mathbf{m}\,1 = \mathbf{m} \tag{8}$$

for a general multivector **m**.

Every multivector **m** has a unique additive inverse, i.e., (-**m**):

$$\mathbf{m} + (-\mathbf{m}) = 0. \tag{9}$$

As stressed in References 1 to 3 and in contrast to Gibbs dot and cross products, all non-zero blades and most non-zero multivectors have unique multiplicative inverses with respect to the geometric product. (We illustrate this feature in Section 3.)

3.0 EXAMPLES

This section gives examples of applications of the geometric algebra package to obtain standard results in G(3). Many of these "results" have been built into the code; some are not so obvious. The development of code for inverses might more appropriately be considered part of Section 4, Applications; in

any case, the expressions for multivector inverses are essential parts of the code and are incorporated into the implementation package (Appendix).

3.1 Dot, Cross, and Bivector Products

3.1.0 Define some vectors and pseudovectors

In[72]:=

```
spa = MakeMV[0, a, 0, 0]; spb = MakeMV[0, b, 0, 0];
spB = MakeMV[0, 0, B, 0]; spA = MakeMV[0, 0, A, 0];
```

3.1.1 Dot- (wedge-) product is defined in terms of min (max) grade terms in a GP:

3.1.1.0 For a vector <a> and grade r multivector Br:

$$\langle a\rangle . B_r = (\langle a\rangle B_r - (-1)^r B_r\langle a\rangle)/2 \quad \text{and}$$

$$\langle a\rangle \wedge B_r = (\langle a\rangle B_r + (-1)^r B_r\langle a\rangle)/2$$

The grade of <a>.Br is r-1; the grade of <a>$\wedge$Br is r+1.

3.1.1.1 One could define commutator and anticommutator products of arbitrary multivectors as:

In[76]:=

```
com[a_, b_] := (GP[a, b] - GP[b, a])/2
```

In[77]:=

```
anti[a_, b_] := (GP[a, b] + GP[b, a])/2
```

3.1.2 Vector-vector products:

3.1.2.1 Wedge product: The commutator of vectors yields a pseudovector: com[<a>,<b>] = <a>$\wedge$<b> = i <a>X<b>.

In[78]:=

```
com[spa, spb]
```

Out[78]=

```
0+(0)+i(<a>X<b>)+i(0)
```

3.1.2.2 Dot product: The anticommutator of vectors yields a scalar: anti[<a>,<b>] = <a>.<b> = (a,b).

```
In[79]:=
      anti[spa, spb]
Out[79]=
     (a,b)+(0)+i(0)+i(0)
```

3.1.3 Vector-pseudovector products:

3.1.3.1 Dot product: Commutator of vector and a pseudovector yields a vector: com[<a>,i <A>] = i <a>∧<A> = -<a>X<A>.

```
In[80]:=
      com[spa, spA]
Out[80]=
     0+(-<a>X<A>)+i(0)+i(0)
```

3.1.3.2 Cross (wedge) product: The anticommutator of a vector and a pseudovector yields a pseudoscalar:

anti[<a>,i<A>] = i <a>.<A> = i (a,A).

```
In[81]:=
      anti[spa, spA]
Out[81]=
     0+(0)+i(0)+i((a,A))
```

3.1.4.1 The commutator product of pseudovectors yields a pseudovector. It is neither a dot nor a cross product of pseudovectors; its grade is 2. (The wedge product would be grade 4; thus, it is 0.) The commutator product of bivectors is given by -1 times the cross (wedge) product of their dual vectors:

com[i<A>,i<B>] = -com[<A>,<B>] = -<A>∧<B> = -i <A>X<B>.

```
In[82]:=
      com[spA, spB]
Out[82]=
      -(A,B)+(0)+i(-<A>X<B>)+i(0)
```

3.1.4.2 Exercise: Multivectors of the form a_0 + i<q>, where a_0 is a scalar and <q> is a vector, are called spinors. Show that spinors form a 4-dimensional subalgebra of G(3).

3.2 Check on Associativity of Geometric Products

3.2.1 Define three arbitrary multivectors.

```
In[83]:=
      spa = MakeMV[a0, a, A, A0]; spb = MakeMV[b0, b, B, B0];
      spc = MakeMV[c0, c, C, C0];
```

3.2.2 To get an idea of what is involved in the geometric product of three multivectors, let's exhibit the vector part of such a product:

```
In[85]:=
      vector[GP[spa, GP[spb, spc]]]
Out[85]=
      b0 c0 <a> - B0 C0 <a> + (b,c) <a> - (B,C) <a> - B0 c0 <A> -
      b0 C0 <A> - (b,C) <A> - (B,c) <A> + a0 c0 <b> - A0 C0 <b> -
      (a,c) <b> + (A,C) <b> - A0 c0 <B> - a0 C0 <B> + (a,C) <B> +
      (A,c) <B> + a0 b0 <c> - A0 B0 <c> + (a,b) <c> - (A,B) <c> -
      A0 b0 <C> - a0 B0 <C> - (a,B) <C> - (A,b) <C> -
      C0 <a>X<b> - c0 <a>X<B> - B0 <a>X<c> - b0 <a>X<C> -
      c0 <A>X<b> + C0 <A>X<B> - b0 <A>X<c> + B0 <A>X<C> -
```

```
A0 <b>X<c> - a0 <b>X<C> - a0 <B>X<c> + A0 <B>X<C>
```

3.2.3 A demonstration that GP[m1,GP[m2,m3]]-GP[GP[m1, m2],m3] = 0.

In[86]:=

```
Timing[GP[spa, GP[spb, spc]] - GP[GP[spa, spb], spc]]
```

Out[86]=

```
{62.56 Second, 0+(0)+i(0)+i(0)}
```

3.3 Elementary Properties of Involuntary Transformations of Products

3.3.1 Exercise: For arbitrary multivectors a and b, show that spatialReversal[a b] = spatialReversal[b] spatialReversal[a].

3.3.2 Exercise: For arbitrary multivectors a and b, show that hermitean[a b] = hermitean[b] hermitean[a].

3.3.3 Exercise: For arbitrary multivectors a and b, show that spatialInversion[a b] = spatialInversion[a] spatialInversion[b].

3.4 Inverse of General Multivectors

3.4.1 Inverses of combinations of scalars and psuedoscalars:

As in the case of complex scalars, for i the unit pseudoscalar, (a0 + i c0)(a0 - i c0) = $a0^2 + c0^2$ is a real non-negative scalar:

In[87]:=

```
GP[MV[a0, 0, 0, c0], MV[a0, 0, 0, -c0]]
```

Out[87]=

```
  2     2
a0  + c0 +(0)+i(0)+i(0)
```

Thus, unless $a0^2+c0^2 = 0$, the inverse of a0 + i c0 is given by:

In[88]:=

```
inverse[MV[a0_, 0, 0, c0_]] :=
                    (1*MV[a0, 0, 0, -c0])/(a0^2 + c0^2)
```

In[89]:=
```
inverse[MV[a0_, 0, 0, 0]] := MakeMV[1/a0, 0, 0, 0]
```

In[90]:=
```
inverse[MV[0, 0, 0, c0_]] := MakeMV[0, 0, 0, -c0^(-1)]
```

3.4.2 Inverses of general elements:

For an arbitrary multivector v,

GP[spatialReversal[v], v] = a "complex" scalar:

In[91]:=
```
v = MakeMV[a0, a, A, A0];
```

In[92]:=
```
GP[spatialReversal[v], v]
```

Out[92]=
```
  2      2
a0  - A0  - (a,a) + (A,A)+(0)+i(0)+i(2 (a0 A0 - (a,A)))
```

Thus, unless the "complex" scalar,

GP[spatialReversal[v], v] vanishes, the inverse of an arbitrary

multivector v exists and is given by:

In[93]:=
```
inverse[x_MV] := Module[{rx = spatialReversal[x]},

     GP[inverse[Chop[GP[x,rx]]],rx]]
```

(We include Chop to handle numerical cases.)

3.4.3 Example: Inverse of a spinor is a spinor:

In[94]:=
```
inversespa = inverse[spa = MakeMV[a0, 0, A, 0]]
```

Out[94]=
```
    a0                      <A>
-----------+(0)+i(-(-----------))+i(0)
  2                    2
a0  + (A,A)          a0  + (A,A)
```
In[95]:=
```
GP[inversespa, spa]
```

```
Out[95]=
     1+(0)+i(0)+i(0)
In[96]:=
      GP[spa, inversespa]
Out[96]=
     1+(0)+i(0)+i(0)
```

4.0 APPLICATIONS

This section shows how the implementation of the geometric algebra of G(3) can be used to obtain the solution of multivector equations and to develop an algebraic treatment (without matrices) of rotations in R(3). The choice of applications is, of course, arbitrary.

4.1 Solution of Multivector Equations

4.1.0.1 Hestenes suggests elegant techniques for solving multivector equations, which involve replacing dot and wedge (i.e., cross) products by appropriate combinations of geometric products so as to convert to multivector equations to the form m1 <x> = m2 and then applying the inverse of m1.

4.1.0.2 Here we approach the same problems by a more "brute force" technique, viz., (1) Get equation to be solved in multivector form. (2) Step 1. Form geometric products with the vectors and/or (duals to the) pseudovectors in the problem. Solve the various grade terms for the dot and wedge (cross) products to be eliminated. (We use the fact that if m=0, where m is a multivector, then b m = 0, where b is an arbitrary

multivector.) (3) Step 2. Plug in the dot and wedge products and solve for x.

4.1.1 Hestenes 2-1 Exercises (1.3)

Solve alpha **<x>** + **<x>.<b>** **<a>** = **<c>** for **<x>.**

Get lhs of equation to be solved in multivector form. Use MVscalar[v1,v2] to get dot product of vectors, etc. (The anticommutator product would also yield the dot product of vectors.)

```
In[97]:=
      spx = MakeMV[0, x, 0, 0]; spa = MakeMV[0, a, 0, 0];
      spb = MakeMV[0, b, 0, 0]; spc = MakeMV[0, c, 0, 0];
In[100]:=
      lhs = alpha*spx + GP[spa, MVscalar[GP[spx, spb]]] - spc
Out[100]=
     0+((b,x) <a> - <c> + alpha <x>)+i(0)+i(0)
```

Step 1. Form geometric product with spb and solve for b.x.

```
In[101]:=
      lhsTimesb = GP[lhs, spb]
Out[101]=
     -(b,c) + alpha (b,x) + (a,b) (b,x)+(0)+
     i((b,x) <a>X<b> + <b>X<c> - alpha <b>X<x>)+i(0)
In[102]:=
      bDotxEq=Solve[scalar[lhsTimesb]==0, dot[vec[b], vec[x]]]
Out[102]=
                          (b,c)
     {{(b,x) -> -------------}}
                  alpha + (a,b)
```

Step 2. Plug b.x into lhs and solve for <x>.

```
In[103]:=

      soln = Solve[(vector[lhs] /. bDotxEq) == 0, vec[x]]

Out[103]=
                 (b,c) <a> - alpha <c> - (a,b) <c>
     {{<x> -> -(---------------------------------)}}
                             2
                        alpha  + alpha (a,b)
```

Collect and Simplify the term proportional to vec[c].

```
In[104]:=
      MapAt[Simplify[Collect[#1, vec[c]]] & , soln, {1, 1, 2}]

Out[104]=
                     (b,c) <a>              <c>
     {{<x> -> -(--------------------) + -----}}
                      2                 alpha
                 alpha  + alpha (a,b)
```

4.1.2 Hestenes 2-1 Exercises (1.4)

Solve alpha <x> + <x>.(i<B>) = <c> for <x>.

Get lhs of equation in multivector form. Use dual form for the bivector, i.e., <B> is a vector and i<B> is the bivector. (Remember that <x>.Bivector is the vector part of GP[x,Bivector].)

```
In[105]:=

      spx = MakeMV[0, x, 0, 0]; spa = MakeMV[0, a, 0, 0];

In[107]:=

      spB = MakeMV[0, 0, B, 0]; spc = MakeMV[0, c, 0, 0];

In[109]:=

      lhs = MVvector[alpha*spx + GP[spx, spB] - spc]

Out[109]=

     0+(-<c> + alpha <x> + <B>X<x>)+i(0)+i(0)
```

Step 1. Eliminate the BXx term. N.b., this entails eliminating the B.x term that appears in GP with spB.

```
In[110]:=
       lhsTimesB = GP[lhs, spB]
Out[110]=
      0+((B,x) <B> - (B,B) <x> - <B>X<c> + alpha <B>X<x>)+i(0)+
      i(-(B,c) + alpha (B,x))
```

Use Thread and make the multivector head (i.e., MV) go to List:

```
In[111]:=
       Thread[lhsTimesB == MV[0, 0, 0, 0], MV] /. MV -> List
Out[111]=
      {True, (B,x) <B> - (B,B) <x> - <B>X<c> + alpha <B>X<x> == 0,
      True, -(B,c) + alpha (B,x) == 0}
```

Solve the equations for <B>.<x> and <B>X<x>:

```
In[112]:=
       eeqs = Solve[Thread[lhsTimesB == MV[0, 0, 0, 0], MV] /.
 MV -> List,{dot[vec[B], vec[x]], vec[Cross[vec[B], vec[x]]]}]
Out[112]=
                   (B,c)
      {{(B,x) -> -----, <B>X<x> ->
                   alpha

           (B,c) <B>     -((B,B) <x>) - <B>X<c>
        -(---------) - ----------------------}}
                2                alpha
           alpha
```

Step 2. Plug in <B>X<x> and <B>.<x> and solve for <x> = vec[x].

To get the bivector forms from the duals use:

$$\texttt{<c>.<B> <B> = -<c>.<B>i (i<B>) = -<c>}\wedge\texttt{(i<B>)(i<B>)}$$

$$\texttt{and <B>X<c> = -i <B>}\wedge\texttt{<c> = -(i<B>).<c>}$$

```
In[113]:=
      soln = Solve[vector[lhs] == 0 /. eeqs[[1]], vec[x]]
```

```
Out[113]=
                                           2
               -((B,c) <B>) - alpha  <c> + alpha <B>X<c>
    {{<x> -> -(-----------------------------------------)}}
                                       3
                               alpha  + alpha (B,B)
```

Invoke the function Simplify.

```
In[114]:=
     MapAt[Simplify, soln, {1, 1, 2}]
```

```
Out[114]=
                                     2
              (B,c) <B> + alpha  <c> - alpha <B>X<c>
    {{<x> -> --------------------------------------}}
                                  3
                            alpha  + alpha (B,B)
```

4.1.3 Hestenes 2-6 Exercises (6.5)

Describe the solution set of the simultaneous equations:

<x>∧(i<A>) = da and <x>∧(i<B>) = db,

where (i<A>)(i<B>)-(i<B>)(i<A>) = -(<A><B>-i<B><A>) = -<A>X<B>

is not zero. (Actually Hestenes takes da = db = 0.)

Define the bivectors (pseudovectors) for the problem:

```
In[115]:=

     spA = MakeMV[0, 0, A, 0]; spB = MakeMV[0, 0, B, 0];
```

Expand the solution to be found in a basis set. By assumption <A>X<B> is not zero, thus, <A>, <B>, <A>X<B> span 3-space and any <x> can be expanded in the form:

```
In[117]:=

     xtest = MakeMV[0, alpha*vec[A] + beta*vec[B] +

                         gamma*vec[Cross[vec[A], vec[B]]], 0, 0]
```

```
Out[117]=

    0+(alpha <A> + beta <B> + gamma <A>X<B>)+i(0)+i(0)
```

The solution must satisfy the wedge product constraints, and

since (vector)∧(bivector) is a pseudoscalar, we can obtain the values for alpha and beta via:

```
In[118]:=
      constraints = MapAt[Simplify,
     Solve[{pseudoS[GP[xtest, spA]] == da,
          pseudoS[GP[xtest, spB]] == db}, {alpha, beta, gamma}],
                         {{1, 1, 2}, {1, 2, 2}}]

Out[118]=
                  -(db (A,B)) + da (B,B)
     {{alpha -> ----------------------,
                        2
                  -(A,B)  + (A,A) (B,B)

                 db (A,A) - da (A,B)
       beta -> ---------------------}}
                       2
                 -(A,B)  + (A,A) (B,B)
```

Note that the constraints put no limits on gamma. Thus, the solution is the line determined in parametric form as a function of gamma. I.e., any gamma will satisfy the constraints, and the solution set corresponds with the straight line intersection of the planes determined by <x>∧(i<A>) = da and <x>∧(i<B>) = db. (The Simplify[Collect[... code is arrived at by experience or in the present case by trial and error.)

```
In[119]:=
      solution = MapAt[Simplify[Collect[#1,
   vec[Cross[vec[A], vec[B]]]]] & , xtest /. constraints, {1, 2}]

Out[119]=
          (-(db (A,B)) + da (B,B)) <A>
     {0+(---------------------------- +
                      2
             -(A,B)  + (A,A) (B,B)
```

```
   (db (A,A) - da (A,B)) <B>
   ------------------------- + gamma <A>X<B>)+i(0)+i(0)}
             2
     -(A,B)  + (A,A) (B,B)
```

Separate the terms that are proportional to da, db, and gamma:

(The [[1]] gets the multivector out of the list.)

```
In[120]:=
      soln = MapAt[MapAt[Together,
             Collect[#1, {da, db, gamma}], {{1}, {2}}] & ,
                            solution, {1, 2}][[1]]
```

```
Out[120]=

     db ((A,B) <A> - (A,A) <B>)   da ((B,B) <A> - (A,B) <B>)
0+(-------------------------- + -------------------------- +
             2                          2
        (A,B)  - (A,A) (B,B)       -(A,B)  + (A,A) (B,B)

  gamma <A>X<B>)+i(0)+i(0)
```

Check to see if constraints are satisfied:

```
In[121]:=

      {pseudoS[GP[soln, spA]]==da, pseudoS[GP[soln, spB]]==db}

Out[121]=

     {True, True}
```

Check to see if constraints were satisfied in the earlier form:

```
In[122]:=

      {pseudoS[GP[solution[[1]], spA]] == da,
                            pseudoS[GP[solution[[1]], spB]] == db}

Out[122]=

     {True, True}
```

4.2 Rotation Operators in 3-Space

4.2.1 Reflection in the (i<a>)-plane.

Demonstrate that -<a><x> inverse[<a>] is <x> reflected in the (i<a>)-plane.

```
In[123]:=
      spa = MakeMV[0, a, 0, 0]; spx = MakeMV[0, x, 0, 0];

In[124]:=
      Apart /@ (-GP[spa, GP[spx, inverse[spa]]])

Out[124]=
          -2 (a,x) <a>
     0+(------------ + <x>)+i(0)+i(0)
             (a,a)
```

In terms of unit normal <ahat> = <a>/a:

```
In[125]:=
      Apart /@ (-GP[spa, GP[spx, inverse[spa]]]) /.
     vec[a] ->a*vec[ahat] /. dot[vec[ahat], vec[ahat]] -> 1

Out[125]=
     0+(-2 (ahat,x) <ahat> + <x>)+i(0)+i(0)
```

Since <x> = (<a>inverse[<a>])<x>

= <a>(inverse[<a>].<x> + inverse[<a>]∧<x>)

= <a> <x>.inverse[<a>] -<a> <x>∧inverse[<a>],

consider -<a> <x>∧inverse[<a>]:

```
In[126]:=
     Apart /@ (-GP[spa, MVpseudoV[GP[spx, inverse[spa]]]])

Out[126]=
            (a,x) <a>
     0+(-(---------) + <x>)+i(0)+i(0)
              (a,a)
```

One sees that -<a> <x>∧inverse[<a>] is the component of <x> in

the (i<a>)-plane, and that <a> <x>.inverse[<a>] is the component of <x> along <a> (i.e., perpendicular to the (i<a>)-plane).

In[127]:=

```
      Apart /@ (-GP[spa, GP[spa, MVpseudoV[GP[spx,
                                        inverse[spa]]]]])
```

Out[127]=

```
     0+(0)+i(<a>X<x>)+i(0)
```

4.2.2 Two reflections are equivalent to a rotation.

4.2.2.1 One can demonstrate that two reflections are equivalent to a rotation by "back of the envelope" constructions. It may be seen that the rotation is through an angle twice that between the normals and about the line of intersection of the reflection planes.

4.2.2.2 One can also use the package to demonstrate that two reflections are equivalent to a rotation for specific cases.

Define a double reflection function:

In[128]:=

```
     doubleR[a_, b_, spx_MV] :=
     Module[{spa = MakeMV[0,a,0,0], spb = MakeMV[0,b,0,0], w},
     CombineMVlist[
          Apart /@ GP[GP[w = GP[spb, spa], spx], inverse[w]]]]
```

4.2.2.3 Example: Let **a** = {0,0,1} and **b** = {Sin[th/2],0,Cos[th/2]} and operate on a general vector <{x,y,z}>. (We use Expand with Trig->True to apply trigonometric identities.)

In[129]:=

```
      spv = MakeMV[0, {x, y, z}, 0, 0];
```

```
In[130]:=
      (ExpandAll[#1, Trig -> True] & ) /@
               doubleR[{0, 0, 1}, {Sin[th/2], 0, Cos[th/2]}, spv]
Out[130]=
     0+(<{x Cos[th] + z Sin[th], y, z Cos[th] - x Sin[th]}>)+
          i(0)+i(0)
```

4.2.3 The rotation operator is a spinor.

4.2.3.1 The doubleR function could be written in the form

<x> -> inverse[R] <x> R where R = <a><b>, a spinor,

for reflections in the (i<a>)-plane followed by a reflection in the (i<b>)-plane. The effect of R does not depend on the magnitude of <a> and <b>. Without loss of generality, we treat the case that <a> and <b> are unit vectors and thus,

inverse[R] = <b><a> = hermitean[<a><b>] = hermitean[R]

and R = <a><b> = (a,b) + i <a>X<b>

= Cos[theta] + (i<w>) Sin[theta],

where theta is the angle between the normal vectors, <a> and <b>, and <w> is a unit vector in the direction of <a>X<b>.

4.2.3.2 Euler form of the rotation operator. Write the spinor R in the form, R = alpha + i<beta>. Then the Euler parameters, alpha = (a,b) and <beta> = <a>X<b>, define the rotation.

4.2.3.3 The Euler parameters are not independent, since

alpha^2 + <beta><beta> = Cos[theta]^2 +Sin[theta]^2 = 1.

4.2.3.3.1 E.g., Reflections in planes having **a** = {0,0,1} and **b** = {Sin[theta],0,Cos[theta]} yields Euler parameters

alpha = <a>.<b> = Cos[theta] and

<beta> = **<a>X<b>** = <{0,Sin[theta],0}>.

```
In[131]:=

      ExpandAll[GP[MakeMV[0, {0, 0, 1}, 0, 0],
     MakeMV[0, {Sin[th/2], 0, Cos[th/2]}, 0, 0]], Trig -> True]

Out[131]=
         th                  th
     Cos[--]+(0)+i(<{0, Sin[--], 0}>)+i(0)
         2                   2
```

4.2.4 Exponential form of the rotation operator: Exponential function of bivector agument. The expression

R = Cos[theta] + (i<w>) Sin[theta],

suggests that one might express R in the form R = Exp[i**<theta>**]
with **<theta>** = theta **<w>**.

4.2.4.1 Multivector power series for Exp[i<a>]. Cos and Sin of vector argument.

4.2.4.1.1 A function to compute integer powers of multivectors.

```
In[132]:=

      GPpower[a_MV, 1] := a

In[133]:=

      GPpower[a_MV, 0] := MV[1, 0, 0, 0]

In[134]:=

      GPpower[a_MV,(n_Integer)?Positive]:=GP[a,GPpower[a,n-1]]
```

4.2.4.1.2 First six terms in the power series for Exp[spa], where spa = i<a> and let <a>.<a> -> a^2 and <a>->a **<ahat>**.

```
In[135]:=

      spa = MV[0, 0, vec[a], 0];

In[136]:=

      (Collect[Expand[#1], vec[ahat]] & ) /@
```

```
(Sum[GPpower[MakeMV[0, 0, a, 0], i]/i!, {i, 0, 6}] /.
     {dot[vec[a], vec[a]] -> a^2, vec[a] -> a*vec[ahat]})
```

```
Out[136]=
         2    4    6              3    5
        a    a    a              a    a
   1 - -- + -- - ---+(0)+i((a - -- + ---) <ahat>)+i(0)
        2    24   720             6    120
```

4.2.4.1.3 First six terms of the power series for Cos and Sin for argument (I a). ComplexExpand treats arguments not explicitly complex as real, etc.)

```
In[137]:=
      ComplexExpand[Normal[Exp[I*a] + O[a]^7]]
Out[137]=
         2    4    6            3    5
        a    a    a            a    a
   1 - -- + -- - --- + I (a - -- + ---)
        2    24   720           6    120
```

4.2.4.1.4 Thus, Exp[i<a>] may be identified with a multivector having the form of a rotation operator:

```
Exp[i<a>] = Cos[a] + (i<a>/a) Sin[a].
```

4.2.5 Exercise. Show that the power series for Exp[<a>] may be related to those for Sinh and Cosh.

```
Does Exp[i<a>] Exp[<b>] = Exp[<b>] Exp[i<a>]?

Does Exp[i<a>] Exp[i<b>] = Exp[i<b>] Exp[i<a>]?
```

(Forms for Exp[m], where m is a general multivector, are useful in relativity theory (ref 3).)

4.2.6 Rotation operators in exponential form: Exponential function for pseudoscalar argument.

```
In[138]:=
      exp[MV[0, 0, b_, 0]] :=
     Module[{bb = Sqrt[dot[b, b]], bbb, B = 0},
```

```
        bbb = bb /. Sqrt[(wa_)^2] :> wa;
        If[bbb != 0 || !NumberQ[bbb], B += (b*Sin[bbb])/bbb];
        MakeMV[Cos[bbb], 0, B, 0]]
```

```
In[139]:=
    rotation[theta_] :=
    exp[MakeMV[0, 0, theta/2, 0]] /.
         dot[vec[a_], vec[a_]] :> a^2 /.
         {Sqrt[(a_)^2] :> a, 1/Sqrt[(a_)^2] :> 1/a}
```

4.2.6.1 E.g., rotation operator for a rotation thru Abs[theta] about the <theta> axis:

```
In[140]:=
    rotation[theta]
Out[140]=
                              theta
                          Sin[-----] <theta>
         theta                  2
     Cos[-----]+(0)+i(------------------)+i(0)
           2                  theta
```

4.2.6.2 Rotation operator inverse check:

```
In[141]:=
    rotation[th] - inverse[rotation[-th]] /.
                   dot[vec[a_], vec[a_]] :> a^2
Out[141]=
     0+(0)+i(0)+i(0)
```

4.2.7 Identification of the Euler parameters {alpha,beta} of the rotation thru theta:

```
In[142]:=
    erules = Thread[{alpha, 0, beta, 0} -> rotation[theta]/.
                                                     MV ->
List]
```

```
Out[142]=
                                                          theta
                                                      Sin[-----] <theta>
                         theta                              2
    {alpha -> Cos[-----], 0 -> 0, beta -> ------------------,
                           2                                theta

     0 -> 0}
```

4.2.7.1 E.g., rotation of {x,y,z} through theta about {1,0,0} axis, which is easily visualized, etc.

```
In[143]:=
      Timing[Simplify /@ (Expand[vector[
GP[GP[ee = rotation[theta*{1,0,0}], MakeMV[0, {x, y, z}, 0, 0]],
                inverse[ee]]], Trig -> True] /. vec[a_] :> a)]

Out[143]=
     {8.02 Second, {x, y Cos[theta] + z Sin[theta],
            z Cos[theta] - y Sin[theta]}}
```

4.2.7.2 E.g.,rotation of {x,y,z} through th about axis {1,1,0} and turn it back. (To see the turned vector, remove the semicolon.)

```
In[144]:=
      rturned = MV[0, vec[(Collect[Simplify[#1],{x,y,z}]&) /@
(Expand[vector[
     GP[GP[ee = rotation[(th*{1, 1, 0})/Sqrt[2]],
     MakeMV[0,{x,y,z},0,0]], inverse[ee]]], Trig -> True] /.
                vec[a_] :> a)       ], 0, 0];
```

Turn the rotated vector back:

```
In[145]:=
      Timing[Simplify /@
(Expand[vector[GP[GP[inverse[ee],rturned], ee]], Trig -> True] /.
```

```
                    vec[a_] :> a)        ]
```

```
Out[145]=
     {95.9 Second, {x, y, z}}
```

4.2.8 Composition of rotations:

4.2.8.1 The Product of exponential forms: rotation(**<th1>**) followed by rotation(**<th2>**).

To neaten up the notation, let dot[vec[a],vec[a]]->a^2 and choose the positive branch of the Sqrt[a^2].

```
In[146]:=
      rotProd = Apart /@
                    (GP[rotation[th1], rotation[th2]] /.
                    {dot[vec[a_], vec[a_]] :> a^2} //.
                    {Sqrt[(b_)^2] :> b, 1/Sqrt[(c_)^2] :> 1/c}   )
```

```
Out[146]=
                                              th1      th2
                                 (th1,th2) Sin[---] Sin[---]
        th1      th2                            2        2
     Cos[---] Cos[---] - ---------------------------+(0)+i(
         2        2                   th1 th2

                th2      th1                      th1      th2
      th2 Cos[---] Sin[---] <th1> + th1 Cos[---] Sin[---] <th2>
               2        2                         2        2
      ---------------------------------------------------------- -
                              th1 th2

          th1      th2
       Sin[---] Sin[---] <th1>X<th2>
           2        2
       ---------------------------)+i(0)
                th1 th2
```

4.2.8.2 The expression is familiar in terms of unit vectors.

I.e., Let <hati> = <thi>/Abs[thi]:

```
In[147]:=
      rotPprod2 = MapAt[Expand[#1] & , rotProd /.

            {vec[th1] -> th1*vec[hat1],

             vec[th2] -> vec[hat2]*th2}, {{1}, {3}}]
Out[147]=

        th1       th2                            th1       th2
   Cos[---] Cos[---] - (hat1,hat2) Sin[---] Sin[---]+(0)+i(
         2         2                              2         2

         th2       th1                     th1       th2
    Cos[---] Sin[---] <hat1> + Cos[---] Sin[---] <hat2> -
          2         2                       2         2

          th1       th2
     Sin[---] Sin[---] <hat1>X<hat2>)+i(0)
           2         2
```

4.2.8.3 Special case. Rotations about the same axis:

```
In[148]:=
      MapAt[Expand[#1, Trig -> True] & , rotProd2 /.

            hat2 -> hat1 /. dot[vec[a_], vec[a_]] :> a^2 /.

            hat1^2 -> 1, {{1}, {3}}]

Out[148]=

        th1   th2                    th1   th2
   Cos[--- + ---]+(0)+i(Sin[--- + ---] <hat1>)+i(0)
         2     2                      2     2
```

4.2.8.4 Special case. Rotation by Pi (reflection) about the x axis followed by same about y axis:

```
In[149]:=

  pi=N[Pi];r2=Chop[GP[rotation[pi*{1,0,0}],rotation[pi*{0,1,0}]]]

Out[150]=

     0+(0)+i(-<{0, 0, 1.}>)+i(0)
```

4.2.8.5 Special case. Rotation by Pi/2 about the x axis followed by same about y axis. Use CombineMVlist to do vector sums, etc.

In[151]:=

```
CombineMVlist[GP[rotation[(pi*{1, 0, 0})/2],
                 rotation[(pi*{0, 1, 0})/2]]]
```

Out[151]=

```
0.5+(0)+i(<{0.5, 0.5, -0.5}>)+i(0)
```

I.e., Pi/2 about y followed by Pi/2 about x yields Pi/3 = 60 degree rotation around {1,1,-1}.

4.2.9 The Product of Euler spinor forms. Euler spinor for composition of rotations expressed as (geometric) product Euler spinors. A derivation of Hestenes (ref 1), Eqns 3.28.

In[152]:=

```
Thread[MakeMV[alpha, 0, beta, 0] ==
 GP[MakeMV[alpha1,0,beta1,0],MakeMV[alpha2, 0, beta2, 0]], MV] /.
              MV -> List
```

Out[152]=

```
{alpha == alpha1 alpha2 - (beta1,beta2), True,
 <beta> == alpha2 <beta1> + alpha1 <beta2> -
   <beta1>X<beta2>, True}
```

REFERENCES

1. D. Hestenes, New Foundations for Classical Mechanics, Reidel, Dordrecht, 1987.

2. For example, D. Hestenes, "Real Spinor Fields," J. Math. Phys., Vol. 8, 1967, pp. 798-808; D. Hestenes, "Multivector Calculus," J. Math. Anal. and Appl., Vol. 24, 1968, pp. 313-325; D. Hestenes, "Multivector Functions," J. Math. Anal. and Appl., Vol. 24, 1968, pp. 467-473; D. Hestenes, "Vectors, Spinors, and Complex Numbers in Classical and Quantum Physics," Am. J. Phys., Vol. 39, 1971, p. 1013; and D. Hestenes, "Observables, Operators, and Complex Numbers in the Dirac Theory," J. Math. Phys., Vol. 16, 1975, pp. 556-572.

3. W.E. Baylis, J. Huschilt, and Jiansu Wei, "Why i?" Am. J. Phys., Vol. 60, 1992, pp. 788-797.

APPENDIX: THE PACKAGE

```
(* MV is a package for performing operations in the 8-dimensional
        geometric algebra G(3).

         Author: Lawrence V. Meisel
         Version of November 1992. *)

BeginPackage["LM`MV`"]

(* usage statements for the exported functions. *)
MakeMV::usage =
"MakeMV[p0,p,q,q0] constructs a representation, \n
     MV[p0,vec[p],vec[q],q0], of\n
            p0 + <p> + i<q> + i q0.              \n
The package recognizes that                      \n
    i. objects with the head MV are multivectors\n
    ii. objects with head vec are vectors.          \n
                                       \n
See GP for forming geometric products of multivectors.   \n
See scalar, vector, pseudoV, and pseudoS for\n
    selecting parts of multivectors.            \n
See MVscalar, MVvector, MVpseudoV, and MVpseudoS for      \n
    creating MV with the selected multivector parts,      \n
        E.g. MVvector[MV[a0,a,b,b0]]->MV[0,a,0,0]."

GP::usage =
"GP[mv1,mv2] computes the geometric product of the\n
multivectors mv1 and mv2, which must have head MV.  \n
Linear combinations use standard +, etc.              \n
E.g. GP[MakeMV[p0,0,q,0],mv1] + 3 GP[mv2,mv3] yields     \n
     the multivector: (p0 + i<q>) mv1 +3 mv2 mv3. \n
     See also MakeMV."

listCombineMV::usage =
"listCombineMV[mv] simplifies MV's having List-form vectors and
pseudovectors."

Cross::usage = "In G(3): a b = dot(a,b) + a/\\b \n
          \t\t\t               dot(a,b) + i Cross[a,b]."

dot::usage = "In G(3): a b = dot(a,b) + i Cross[a,b]."

vec::usage = "vec[a] means that a is of type vec, i.e., vector."

MV::usage = "MV[mv] means that mv is of type MV, \n
\t \t \t i.e., a multivector."

scalar::usage =
"scalar[MV[a, b, c, d]] returns a, the scalar part\n
     of its multivector argument.\n
```

```
See also MakeMV and MVscalar"

vector::usage =
"vector[MV[a, b, c, d]] returns b, the vector part \n
     of its multivector argument. \n

See also MakeMV and MVvector."

pseudoV::usage =
"pseudoV[MV[a, b, c, d]] returns c, the vector dual\n
to the pseudovector part of its multivector argument.\n

See also MakeMV and MVpseudoV."

pseudoS::usage =
"pseudoS[MV[a, b, c, d]] returns d, the pseudoscalar\n
     part of its multivector argument.\n

See also MakeMV and MVpseudoS"

MVscalar::usage =
"MVscalar[MV[a, b, c, d]] returns MV[a,0,0,0] the pure\n
scalar part of its multivector argument.\n

See also MakeMV and scalar"

MVvector::usage =
"MVvector[MV[a, b, c, d]] returns MV[0,b,0,0] the pure
vector part of its multivector argument. \n

See also MakeMV and vector."

MVpseudoV::usage =
"MVpseudoV[MV[a, b, c, d]] returns MV[0,0,c,0], the     \n
the pure pseudovector part of its multivector argument.\n

See also MakeMV and pseudoV."

MVpseudoS::usage =
"MVpseudoS[MV[a, b, c, d]] returns d, the pure\n
pseudoscalar part of its multivector argument.\n

See also MakeMV and pseudoS."

hermitean::usage =
"hermitean[MV[a0,a,b,b0]] --> MV[a0, a, -b, -b0], \n
     i.e., hermitean[a0 + <a> + i<b> + i b0] ->\n
          a0 + <a> - i<b> - i b0.  \n

See also spatialReversal, spatialInversion, and MakeMV."

spatialReversal::usage =
```

```
"spatialReversal[MV[a0,a,b,b0]] --> MV[a0, -a, -b, b0], \n
     i.e., hermitean[a0 + <a> + i<b> + i b0] ->    \n
          a0 - <a> - i<b> + i b0.    \n

See also hermitean, spatialInversion, and MakeMV."

spatialInversion::usage =
"spatialInversion[MV[a0,a,b,b0]] --> MV[a0, -a, b, -b0], \n
     i.e., hermitean[a0 + <a> + i<b> + i b0] -> \n
          a0 - <a> + i<b> - i b0.                   \n

See also hermitean, spatialReversal, and MakeMV."

inverse::usage = "inverse[mv] returns the GP inverse of mv."

rotation::usage = "rotation[<theta>] -> rotation mv w.r.t.
theta."

exp::usage =
"exp[a0,a,A,A0]=exp[MakeMV[a0,a,A,A0]]=mv exponential function."

Begin["`Private`"]

MakeMV[p0_,p_,q_,q0_]:=MV[p0,vec[p],vec[q],q0]

(* Define the data type vec for vectors and the duals of
bivectors.
*)
vec[k_?NumberQ l_]:=k vec[l]
vec[vec[a_]]:=vec[a]
vec[a_+b_]:=vec[a]+vec[b]
vec[a_vec c_]:=a c
vec[-a_]:=-vec[a]
vec/:vec[x_Cross y_]:=y vec[x]
vec[0]=0;
dot[a_,0]:=0;Cross[a_,0]:=0;dot[0,a_]:=0;Cross[0,a_]:=0;

(* Properties of Gibbs cross products: *)
Cross/:Cross[a_vec,vec[Cross[b_vec,c_vec]]]:=
     b dot[a,c]-c dot[a,b]
Cross/:Cross[vec[Cross[a_vec,b_vec],c_vec]]:=
     b dot[a,c]-a dot[c,b]
Cross/:Cross[a_vec,Cross[b_vec,c_vec]]:=
     b dot[a,c]-c dot[a,b]
Cross/:Cross[Cross[a_vec,b_vec],c_vec]:=
     b dot[a,c]-a dot[c,b]
Cross/:Cross[a_vec,b_vec]:=-Cross[b,a]/;!OrderedQ[{a,b}]
Cross/:Cross[a_+w_,b_]:=Cross[a,b]+Cross[w,b]
Cross/:Cross[a_,b_+w_]:=Cross[a,b]+Cross[a,w]
Cross/:Cross[-a_,b_]:=-Cross[a,b]
Cross/:Cross[a_,-b_]:=-Cross[a,b]
Cross/:Cross[a_,a_]:=0
```

```
Cross/:Cross[w_ a_vec,b_]:=Cross[a,b]w
Cross/:Cross[a_,w_ b_vec]:=Cross[a,b]w

(*  Properties of Gibbs dot product and combinations
 of dot and cross.  *)
dot/:dot[a_vec,b_vec]:=dot[b,a]/;!OrderedQ[{a,b}]
dot/:dot[a_vec w_,b_]:=w dot[b,a]
dot/:dot[a_,b_vec w_]:=w dot[b,a]
dot/:dot[a_,b_+w_]:= dot[b,a]+dot[a,w]
dot/:dot[a_+w_,b_]:=dot[b,a]+dot[b,w]
dot/:dot[-a_,b_]:=-dot[b,a]
dot/:dot[a_,-b_]:=-dot[b,a]
dot/:dot[a_vec,vec[Cross[b_vec,c_vec]]]:=
    Module[{u,v,w},{u,v,w}=Sort[{a,b,c}];
      Signature[{a,b,c}] dot[u,vec[Cross[v,w]]]]/;
                       !OrderedQ[{a,b,c}]
dot/:dot[vec[Cross[b_vec,c_vec]],a_vec]:=
          dot[b,vec[Cross[c,a]]]
dot[vec[Cross[a_vec,b_vec]],vec[Cross[A_vec,B_vec]]]:=
       dot[a,A]dot[b,B]-dot[a,B]dot[A,b]
dot[a_vec,vec[Cross[b_vec,a_vec]]]:=0
dot[a_vec,vec[Cross[a_vec,b_vec]]]:=0

(* Define a function to apply to simplify all MV
combinations. The present simplification choice
allows one to simplify expressions involving
Cos[x]^2+Sin[x]^2.  Note that this could be
accomplished by setting Trig->True in ExpandAll, but
that entails other transformations, which might be
undesirable. *)
regg[a_]:=Map[Factor[ExpandAll[#/.
          Cos[x_]^2:>(1-Sin[x]^2)]]&,a]

(* Linear Combinations of MV's: *)
MV/:MV[a_,b_,c_,d_]+MV[A_,B_,C_,D_] :=
          MV[a+A,b+B,c+C,d+D]//regg;
MV/:MV[a_,b_,c_,d_]-MV[A_,B_,C_,D_] :=
          MV[a-A,b-B,c-C,d-D]//regg;
MV/:w_*MV[a_,b_,c_,d_]:=MV[w a,w b,w c,w d]//regg;

(* Derivatives of MV's: *)
MV/:D[MV[a_,b_,c_,d_],l_]:=
          MV[D[a,l],D[b,l],D[c,l],D[d,l]];
vec/:D[vec[a_],l_]:=vec[D[a,l]];

(*  Geometric products. *)
GP[MV[a_,0,0,0],MV[A_,B_,Q_,Q0_]]:=
          MV[a A,a B,a Q,a Q0]//regg;
GP[MV[0,0,0,a_],MV[A_,B_,Q_,Q0_]]:=
          MV[-a Q0,-a Q,a B,a A]//regg;
GP[MV[0,b_,0,0],MV[A_,B_,Q_,Q0_]]:=
     MV[dot[b,B],A b-vec[Cross[b,Q]],
```

```
            Q0 b+vec[Cross[b,B]],dot[b,Q]]//regg;
GP[MV[0,0,b_,0],MV[A_,B_,Q_,Q0_]]:=
     GP[MV[0,0,0,1],GP[MV[0,b,0,0],MV[A,B,Q,Q0]]]//regg;
GP[MV[a_,b_,c_,d_],MV[A_,B_,Q_,Q0_]]:=
     (GP[MV[a,0,0,0],MV[A,B,Q,Q0]]+
      GP[MV[0,b,0,0],MV[A,B,Q,Q0]]+
      GP[MV[0,0,c,0],MV[A,B,Q,Q0]]+
      GP[MV[0,0,0,d],MV[A,B,Q,Q0]])//regg;

(*  functions for selecting the parts of multivectors. *)
scalar[MV[a_,b_,c_,d_]]:=a
vector[MV[a_,b_,c_,d_]]:=b
pseudoV[MV[a_,b_,c_,d_]]:=c
pseudoS[MV[a_,b_,c_,d_]]:=d

(*  functions for selecting pure multivector parts of
general multivectors.  *)
MVscalar[MV[a_,b_,c_,d_]]:=MakeMV[a,0,0,0]
MVvector[MV[a_,b_,c_,d_]]:=MakeMV[0,b,0,0]
MVpseudoV[MV[a_,b_,c_,d_]]:=MakeMV[0,0,c,0]
MVpseudoS[MV[a_,b_,c_,d_]]:=MakeMV[0,0,0,d]

(*  Involuntary transformations. *)
hermitean[MV[a0_,a_,b_,b0_]]:=MV[a0,a,-b,-b0]
spatialReversal[MV[a0_,a_,b_,b0_]]:=MV[a0,-a,-b,b0]
spatialInversion[MV[a0_,a_,b_,b0_]]:=MV[a0,-a,b,-b0]

(*Special code for processing list form vectors and
pseudovectors.  *)
Cross[vec[a_List],vec[b_List]]:=CROSS[a,b]
Cross[a_List,b_List]:=CROSS[a,b]
CROSS[{a_,b_,c_},{A_,B_,C_}]:={b C-c B,c A-a C,a B-b A}
dot[vec[a_List],vec[b_List]]:=a.b
dot[l_List,m_List]:=l.m
vec[k_ l_List]:=k vec[l]
vec[{0,0,0}]=0;
listCombineMV=MakeMV[Expand[scalar[#]],
          Expand[vector[#]/.vec[a_]:>a],
          Expand[pseudoV[#]/.vec[a_]:>a],
          Expand[pseudoS[#]]  ]&;

(*  formatting scheme for multivectors:  *)
Format[vec[a_]]:=SequenceForm["<",a,">"]
Format[dot[vec[a_],vec[b_]]]:=
          SequenceForm["(",a,",",b,")"]
Format[Cross[a_vec,b_vec]]:=SequenceForm[a,"X",b]
Format[vec[Cross[a_vec,b_vec]]]:=
          SequenceForm[a,"X",b]
Format[MV[a0_,a_,b_,b0_]]:=
     SequenceForm[a0,"+(",a,")+i(",b,")+i(",b0,")"]

(*  Inverses. *)
```

```
inverse[MV[a0_,0,0,c0_]]:=1/(a0^2+c0^2)MV[a0,0,0,-c0]
(* inverse[MV[a0_,0,0,0]]:=MakeMV[1/a0,0,0,0]
inverse[MV[0,0,0,c0_]]:=MakeMV[0,0,0,-1/c0] *)
inverse[x_MV]:=Module[{rx=spatialReversal[x]},
     (* Include Chop to eliminate vestigial non-zero
      vector and bivector parts in numerical cases.*)
     GP[inverse[Chop[GP[x,rx]]],rx]  ]

(* Exponential functions and the rotation operator in R(3). *)
exp[a0_,a_,b_,b0_]:=exp[MV[a0,vec[a],vec[b],b0]]
exp[MV[a0_,0,0,b0_]]:=
     GP[MV[Exp[a0],0,0,0],MV[Cos[b0],0,0,Sin[b0]]]
exp[MV[0,b_,0,0]]:=
Module[{bb=Sqrt[dot[b,b]],bbb,B=0},
     bbb=bb/.dot[vec[aa_],vec[aa_]]:>aa^2
          /.Sqrt[wa_^2]:>wa;
     If[bbb !=0||!NumberQ[bbb],B+=b Sinh[bbb]/bbb];
     MakeMV[Cosh[bbb],B,0,0]]
exp[MV[0,0,b_,0]]:=
Module[{bb=Sqrt[dot[b,b]],bbb,B=0},
     bbb=bb/.dot[vec[aa_],vec[aa_]]:>aa^2
          /.Sqrt[wa_^2]:>wa;
     If[bbb !=0||!NumberQ[bbb],B+=b Sin[bbb]/bbb];
     MakeMV[Cos[bbb],0,B,0]]
rotation[theta_]:=exp[MakeMV[0,0,theta/2,0]]/.
     dot[vec[a_],vec[a_]]:>a^2 /.
     {Sqrt[a_^2]:>a,1/Sqrt[a_^2]:>1/a}
exp[MV[a0_,a_,b_,b0_]]:=
     GP[GP[GP[MV[Exp[a0],0,0,0],MV[Cos[b0],0,0,Sin[b0]]],
          exp[MV[0,a,0,0]]],exp[MV[0,0,b,0]]]

End[] (* end private context  *)

EndPackage[]  (* end package context *)
/
```

TECHNICAL REPORT INTERNAL DISTRIBUTION LIST

	NO. OF COPIES
CHIEF, DEVELOPMENT ENGINEERING DIVISION	
ATTN: AMSTA-AR-CCB-DA	1
-DB	1
-DC	1
-DD	1
-DE	1
CHIEF, ENGINEERING DIVISION	
ATTN: AMSTA-AR-CCB-E	1
-EA	1
-EB	1
-EC	
CHIEF, TECHNOLOGY DIVISION	
ATTN: AMSTA-AR-CCB-T	2
-TA	1
-TB	1
-TC	1
TECHNICAL LIBRARY	
ATTN: AMSTA-AR-CCB-O	5
TECHNICAL PUBLICATIONS & EDITING SECTION	
ATTN: AMSTA-AR-CCB-O	3
OPERATIONS DIRECTORATE	
ATTN: SMCWV-ODP-P	1
DIRECTOR, PROCUREMENT & CONTRACTING DIRECTORATE	
ATTN: SMCWV-PP	1
DIRECTOR, PRODUCT ASSURANCE & TEST DIRECTORATE	
ATTN: SMCWV-QA	1

NOTE: PLEASE NOTIFY DIRECTOR, BENÉT LABORATORIES, ATTN: AMSTA-AR-CCB-O OF ADDRESS CHANGES.

TECHNICAL REPORT EXTERNAL DISTRIBUTION LIST

	NO. OF COPIES
ASST SEC OF THE ARMY RESEARCH AND DEVELOPMENT ATTN: DEPT FOR SCI AND TECH THE PENTAGON WASHINGTON, D.C. 20310-0103	1
ADMINISTRATOR DEFENSE TECHNICAL INFO CENTER ATTN: DTIC-OCP (ACQUISITION GROUP) BLDG. 5, CAMERON STATION ALEXANDRIA, VA 22304-6145	2
COMMANDER U.S. ARMY ARDEC ATTN: SMCAR-AEE	1
SMCAR-AES, BLDG. 321	1
SMCAR-AET-O, BLDG. 351N	1
SMCAR-FSA	1
SMCAR-FSM-E	1
SMCAR-FSS-D, BLDG. 94	1
SMCAR-IMI-I, (STINFO) BLDG. 59 PICATINNY ARSENAL, NJ 07806-5000	2
DIRECTOR U.S. ARMY RESEARCH LABORATORY ATTN: AMSRL-DD-T, BLDG. 305 ABERDEEN PROVING GROUND, MD 21005-5066	1
DIRECTOR U.S. ARMY RESEARCH LABORATORY ATTN: AMSRL-WT-PD (DR. B. BURNS) ABERDEEN PROVING GROUND, MD 21005-5066	1
DIRECTOR U.S. MATERIEL SYSTEMS ANALYSIS ACTV ATTN: AMXSY-MP ABERDEEN PROVING GROUND, MD 21005-5071	1
COMMANDER ROCK ISLAND ARSENAL ATTN: SMCRI-ENM ROCK ISLAND, IL 61299-5000	1
MIAC/CINDAS PURDUE UNIVERSITY P.O. BOX 2634 WEST LAFAYETTE, IN 47906	1
COMMANDER U.S. ARMY TANK-AUTMV R&D COMMAND ATTN: AMSTA-DDL (TECH LIBRARY) WARREN, MI 48397-5000	1
COMMANDER U.S. MILITARY ACADEMY ATTN: DEPARTMENT OF MECHANICS WEST POINT, NY 10966-1792	1
U.S. ARMY MISSILE COMMAND REDSTONE SCIENTIFIC INFO CENTER ATTN: DOCUMENTS SECTION, BLDG. 4484 REDSTONE ARSENAL, AL 35898-5241	2
COMMANDER U.S. ARMY FOREIGN SCI & TECH CENTER ATTN: DRXST-SD 220 7TH STREET, N.E. CHARLOTTESVILLE, VA 22901	1
COMMANDER U.S. ARMY LABCOM MATERIALS TECHNOLOGY LABORATORY ATTN: SLCMT-IML (TECH LIBRARY) WATERTOWN, MA 02172-0001	2
COMMANDER U.S. ARMY LABCOM, ISA ATTN: SLCIS-IM-TL 2800 POWER MILL ROAD ADELPHI, MD 20783-1145	1

NOTE: PLEASE NOTIFY COMMANDER, ARMAMENT RESEARCH, DEVELOPMENT, AND ENGINEERING CENTER, BENÉT LABORATORIES, CCAC, U.S. ARMY TANK-AUTOMOTIVE AND ARMAMENTS COMMAND, AMSTA-AR-CCB-O, WATERVLIET, NY 12189-4050 OF ADDRESS CHANGES.

TECHNICAL REPORT EXTERNAL DISTRIBUTION LIST (CONT'D)

	NO. OF COPIES
COMMANDER U.S. ARMY RESEARCH OFFICE ATTN: CHIEF, IPO P.O. BOX 12211 RESEARCH TRIANGLE PARK, NC 27709-2211	1
DIRECTOR U.S. NAVAL RESEARCH LABORATORY ATTN: MATERIALS SCI & TECH DIV CODE 26-27 (DOC LIBRARY) WASHINGTON, D.C. 20375	1 1
WRIGHT LABORATORY ARMAMENT DIRECTORATE ATTN: WL/MNM EGLIN AFB, FL 32542-6810	1
WRIGHT LABORATORY ARMAMENT DIRECTORATE ATTN: WL/MNMF EGLIN AFB, FL 32542-6810	1

NOTE: PLEASE NOTIFY COMMANDER, ARMAMENT RESEARCH, DEVELOPMENT, AND ENGINEERING CENTER, BENÉT LABORATORIES, CCAC, U.S. ARMY TANK-AUTOMOTIVE AND ARMAMENTS COMMAND, AMSTA-AR-CCB-O, WATERVLIET, NY 12189-4050 OF ADDRESS CHANGES.